CTSC

CENTER FOR TRUSTWORTHY SCIENTIFIC CYBERINFRASTRUCTURE

The NSF Cybersecurity Center of Excellence

Welcome to the CCoE Webinar Series. Our topic today is Stronger Security for Password Authentication with Stanislaw Jarecki. Our host is Jeannette Dopheide.

The meeting will begin shortly. Participants are muted. You may type questions into the chat box during the presentation.

This meeting will be recorded.

The CTSC Webinar Series is supported by National Science Foundation grant #1547272.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF.
http://hdl.handle.net/2022/21639

Is Password InSecurity Inevitable?

or

How to Strengthen Password and Multi-Factor Authentication

Stanislaw Jarecki (UC Irvine)

Joint works with:

Hugo Krawczyk (IBM Research)

Nitesh Saxena, Maliheh Shirvanian (UA Birmingham)

Aggelos Kiayas (U Edinburgh)

Jiayu Xu (UC Irvine)

Password (In)Security

- Passwords: **MAIN** authentication tool in the digital era
- Protect our lives and social order, *conveniently* and **Insecurely**

Password (In)Security

- Passwords: **MAIN** authentication tool in the digital era
- Protect our lives and social order, *conveniently* and **Insecurely**

Password (In)Security

- Passwords: **MAIN** authentication tool in the digital era
- Protect our lives and social order, *conveniently* and **Insecurely**

Password (In)Security

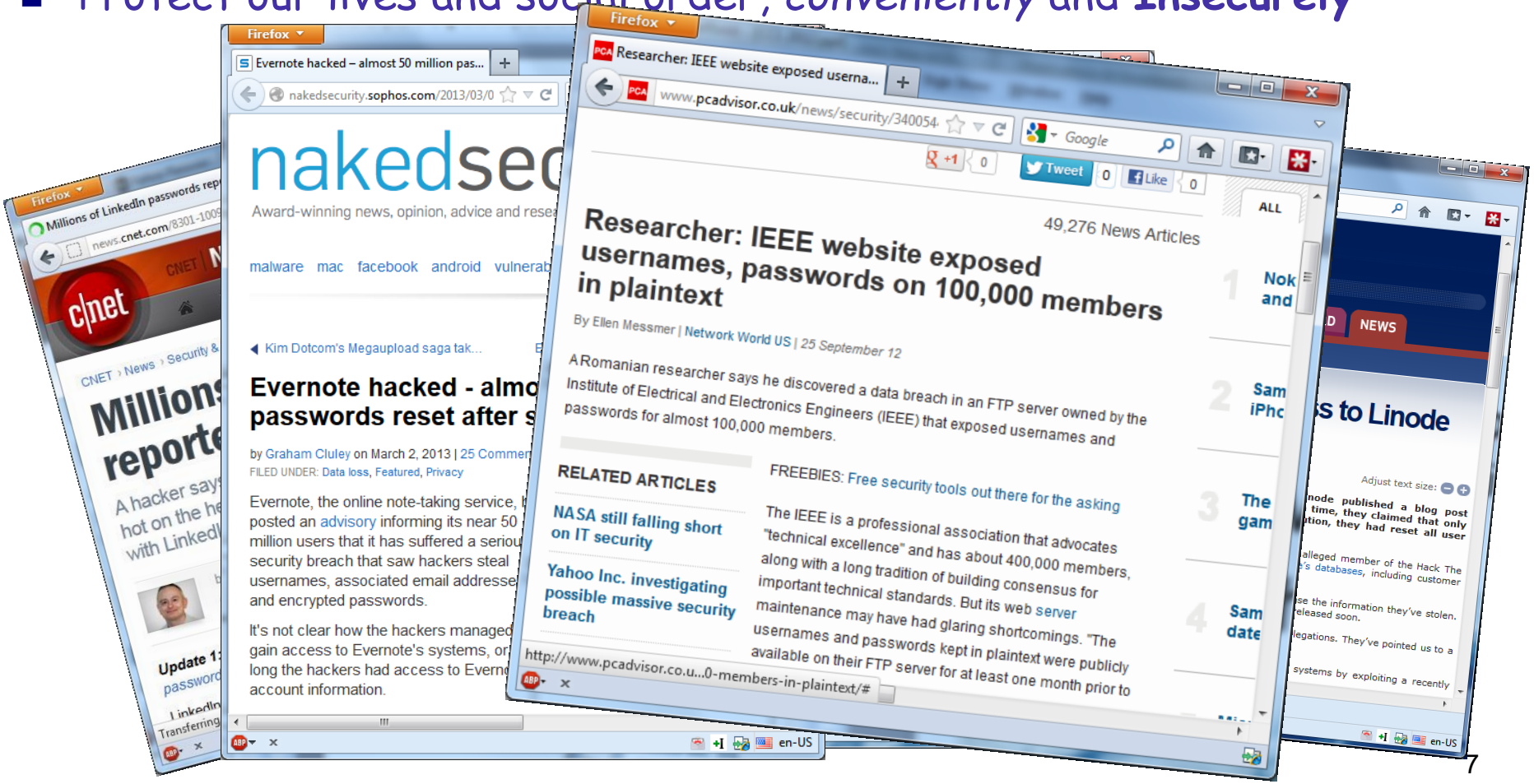
- Passwords: **MAIN** authentication tool in the digital era
- Protect our lives and social order, *conveniently* and **Insecurely**

Password (In)Security

- Passwords: **MAIN** authentication tool in the digital era
- Protect our lives and social order, *conveniently* and **Insecurely**

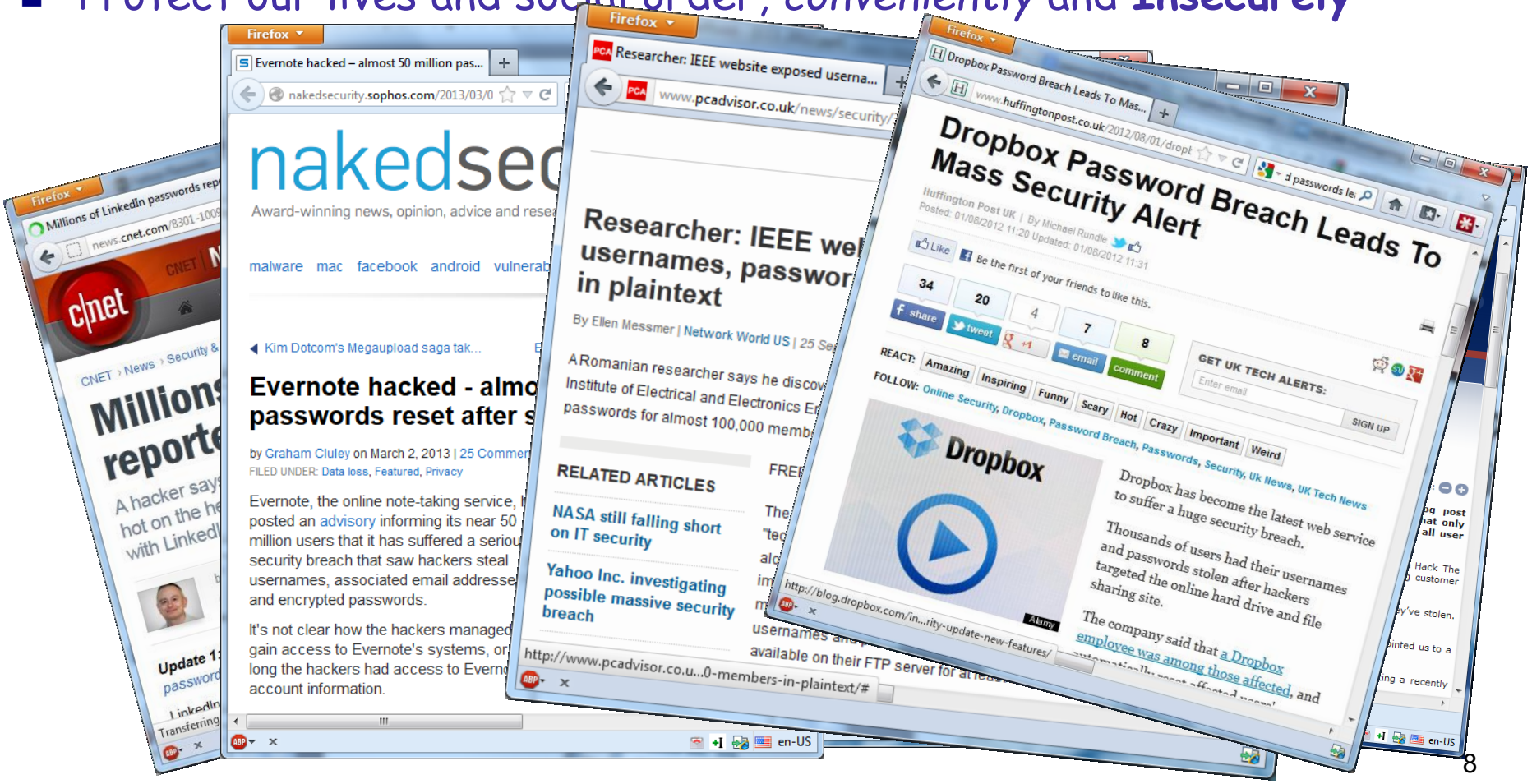
Password (In)Security

- Passwords: **MAIN** authentication tool in the digital era
- Protect our lives and social order, *conveniently* and **Insecurely**



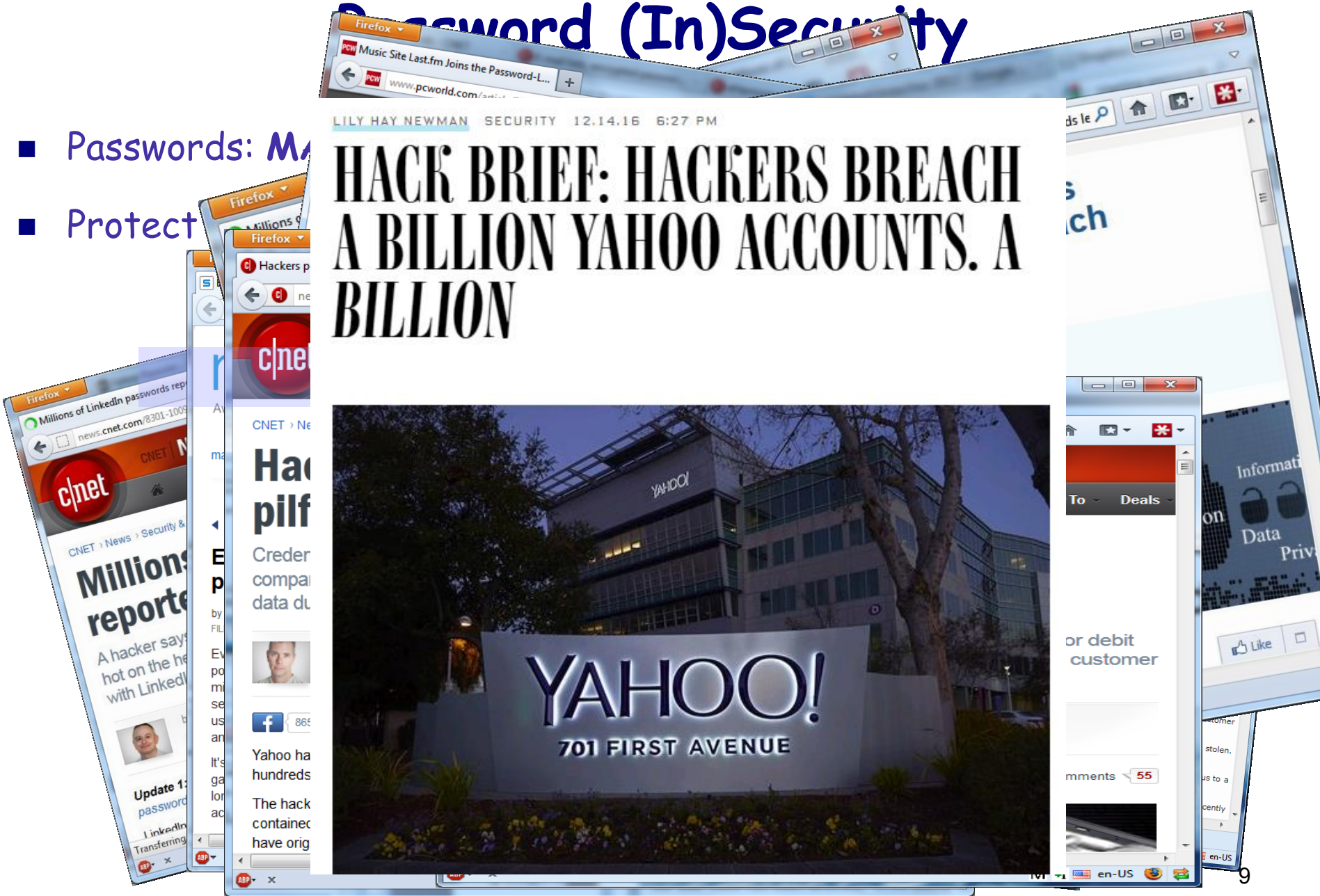
Password (In)Security

- Passwords: **MAIN** authentication tool in the digital era
- Protect our lives and social order, *conveniently* and **Insecurely**



Password (In)Security

- Passwords: Ma
- Protect



Password (In)Security

- Passwords: Ma
- Protect

Password (In)Security

- Passwords: **MAIN** authentication tool in the digital era
- Protect our lives and social order, *conveniently* and **Insecurely**
- BILLIONS of passwords stolen
  - MySpace 360M, LinkedIn 165M, eBay 145M,…, Yahoo 1B (!!)
  - … Twitter, RSA, Google, Dropbox, PayPal, Sony, …
- https://www.leakedsource.com:
  - 2,918,283,623 accounts at your service.
  - "Check for free to see if your email or account was hacked."

Password (In)Security

- Passwords: **MAIN** authentication tool in the digital era
- Protect our lives and social order, *conveniently* and **Insecurely**
- BILLIONS of passwords stolen
  - MySpace 360M, LinkedIn 165M, eBay 145M,…, Yahoo 1B (!!)
  - … Twitter, RSA, Google, Dropbox, PayPal, Sony, …
- https://www.leakedsource.com:
  - 2,918,283,623 accounts at your service.
  - "Check for free to see if your email or account was hacked."

REMOVED

An Unacceptable State of Affairs (but do we have a choice?)

- Unacceptable, really! Our social order depends on passwords
- But do we have a choice?
  - Getting rid of passwords is not realistic: passwords are too convenient, and massively deployed.
  - Ask users to memorize (multiple) high-entropy passwords: No way
  - Stop choosing same/related password: No way

    + it's often less secure: users cannot remember changing passwords, so they get a new password by email every time they authenticate…

Can Cryptography Help?

- **Yes**!
- We show ways to strengthen password protocols for a variety of authentication settings, including common settings used today.
- Using simple, well-established techniques
  - Mostly blinded Diffie-Hellman [Chaum, Ford-Kaliski, Boyen, …] ("oblivious PRF")
- Efficient. Mature. **Ready for deployment** in the **real world**.
- We will go over three such solutions in this talk.
- Pointers to papers at the end; and please talk to me if you are interested to learn more (esp. if you see where we can improve, or if you want to transfer this to practice).

Attacks on Password Authentication #1: Offline Dictionary Attack (ODA)

- ODA is the main source of password compromise:
  - *Deadly combination* of human memory limitation (→low entropy passwords) and server compromise
  - Attacker who gets hold of a "password file" can test candidate passwords against stored hashes; cost proportional to dictionary size
  - Millions++ of passwords tested per second (from s/w to dedicated h/w)

Attacks on Password Authentication #1: Offline Dictionary Attack (ODA)

- ODA is the main source of password compromise:
  - *Deadly combination* of human memory limitation (→low entropy passwords) and server compromise
  - Attacker who gets hold of a "password file" can test candidate passwords against stored hashes; cost proportional to dictionary size
  - Millions++ of passwords tested per second (from s/w to dedicated h/w)

☹ Offline attacks upon server compromise are **unavoidable**

  - If the server can check pwd given password file then so can the attacker
    - Server holds H(pwd) for known deterministic function H (= a hash function)
  - *Password salting* slows the attack but does not eliminate it in practice
    - For each account server holds H(pwd,s) for $ "salt" value s:

      To test pwd, attacker has to compute H(pwd,·) separately for each account

Hope: Render these unavoidable exhaustive attacks ineffective!

How: Enforce high-entropy passwords using additional devices/servers

- What Devices?
  - Cell phone, USB stick: Already used in Two-Factor Authentication
- What Servers?
  - Can be hosted by any cloud service
  - End-users can utilize it *transparently* to web servers
  - Web servers can utilize it *transparently* to end-users

Attacks on Password Authentication #2,3,4,5

2. Online dict. attacks (unavoidable): Guess password; try it online.
   - Works w/weak pwds and in targeted attacks (pers. info, sister pwd) [Wang'16]
3. Phishing attack: User tricked to send password to the wrong server
   - paypa1.com, overwritten links in email, URL-browser manipulation, …
4. PKI attacks: Browser tricked to accept as valid a fake certificate
   - Cert signed by an authorized but rogue CA (do ***you*** know your browser's CA's?)
   - A certificate flagged by the browser but user accepts (by "clicking through")
5. Malware on the client (terminal, laptop, phone), e.g. *keyloggers*

Attacks on Password Authentication #2,3,4,5

2. Online dict. attacks (unavoidable): Guess password; try it online.
   - Works w/weak pwds and in targeted attacks (pers. info, sister pwd) [Wang'16]
3. Phishing attack: User tricked to send password to the wrong server
   - paypa1.com, overwritten links in email, URL-browser manipulation, …
4. PKI attacks: Browser tricked to accept as valid a fake certificate
   - Cert signed by an authorized but rogue CA (do ***you*** know your browser's CA's?)
   - A certificate flagged by the browser but user accepts (by "clicking through")
5. Malware on the client (terminal, laptop, phone), e.g. *keyloggers*

How can we help?

Attacks on Password Authentication #2,3,4,5

✔ (with TFA)

2. Online dict. attacks (unavoidable): Guess password; try it online.
   - Works w/weak pwds and in targeted attacks (pers. info, sister pwd) [Wang'16]
3. Phishing attack: User tricked to send password to the wrong server
   - paypa1.com, overwritten links in email, URL-browser manipulation, …
4. PKI attacks: Browser tricked to accept as valid a fake certificate
   - Cert signed by an authorized but rogue CA (do ***you*** know your browser's CA's?)
   - A certificate flagged by the browser but user accepts (by "clicking through")
5. Malware on the client (terminal, laptop, phone), e.g. *keyloggers*

#2: Can be rendered ineffective if 2nd authentication factor (cell phone, USB stick) used appropriately [our TFA work…]

Attacks on Password Authentication #2,3,4,5

✔ (with TFA)

2. Online dict. attacks (unavoidable): Guess password; try it online.
   - Works w/weak pwds and in targeted attacks (pers. info, sister pwd) [Wang'16]

✔ "Security w/o PKI" (simple crypto + browser extension)

3. Phishing attack: User tricked to send password to the wrong server
   - paypa1.com, overwritten links in email, URL-browser manipulation, …

4. PKI attacks: Browser tricked to accept as valid a fake certificate
   - Cert signed by an authorized but rogue CA (do ***you*** know your browser's CA's?)
   - A certificate flagged by the browser but user accepts (by "clicking through")

 # 3,4 stem from over-reliance on Public Key Infrastructure (PKI):
Client uses wrong PK ⇒ Attacker learns pwd

Attacks on Password Authentication #2,3,4,5

✔ (with TFA)

2. Online dict. attacks (unavoidable): Guess password; try it online.
   - Works w/weak pwds and in targeted attacks (pers. info, sister pwd) [Wang'16]

✔ "Security w/o PKI" (simple crypto + browser extension)

3. Phishing attack: User tricked to send password to the wrong server
   - paypa1.com, overwritten links in email, URL-browser manipulation, …

4. PKI attacks: Browser tricked to accept as valid a fake certificate
   - Cert signed by an authorized but rogue CA (do ***you*** know your browser's CA's?)
   - A certificate flagged by the browser but user accepts (by "clicking through")

 # 3,4 stem from over-reliance on Public Key Infrastructure (PKI): Client uses wrong PK ⇒ Attacker learns pwd

- PAKE's ("Password Auth. Key Exchange") [BPR'00,BMP'00,…,…,…] make Pwd. Auth. secure (up to online attacks) without trust in PKI at cost ≈ TLS/SSL session (≈ 2exp per client/server, 2-3 rounds)
- Our work offers security without PKI (+ no ODA on Server + TFA + …)

Attacks on Password Authentication #2,3,4,5

✔ (with TFA)

2. Online dict. attacks (unavoidable): Guess password; try it online.
   - Works w/weak pwds and in targeted attacks (pers. info, sister pwd) [Wang'16]

✔ "Security w/o PKI" (simple crypto + browser extension)

3. Phishing attack: User tricked to send password to the wrong server
   - paypa1.com, overwritten links in email, URL-browser manipulation, …

4. PKI attacks: Browser tricked to accept as valid a fake certificate
   - Cert signed by an authorized but rogue CA (do ***you*** know your browser's CA's?)
   - A certificate flagged by the browser but user accepts (by "clicking through")

✔ (with TFA)

5. Malware on the client (terminal, laptop, phone), e.g. *keyloggers*

 # 5: Help from 2nd factor (same as #2): Adversary needs to corrupt the password and the 2nd auth. factor (= cell phone, USB, …)

Outline for rest of the talk

1. Sphinx: Improving Password Authentication via Password Store (= auxiliary device / online security server) [JKSS'16,JKSS'17]
   - Extension to Two-Factor Authentication [In submission]

2. Generalized Password-Store: Password-Protected Secret Sharing [BJSL'11, JKK'14, JKKX'16, JKKX'17]

3. XPAKE: Implementing private salt in PKI-free PAKE [In preparation]

Part I:

Removing Offline Dictionary Attacks on Server Compromise

Intuition: Use secure key to securely map:
memorizable password $\rightarrow$ full-entropy passwords
(so servers can safely store their hashes)

= "**Password Store**" security service

Simple solution: Password Store (a.k.a. password manager)

- Carry strong independent passwords:
  - stored … in your phone, your smart watch, …, or retrievable online
  - … encrypted under *a master password*
- Just remember one master password (hopefully non-trivial)

Password store: Not without problems

- A list of user passwords encrypted under the user's master password
  - Attacker obtains the list ➔ Offline Attack against master password (➔ all the user's passwords compromised)
  - Client compromise: Attacker learns master password as user types it
  - User-device communication compromise: master password leaked
  - Furthermore: Typical password managers keep *user-chosen passwords* (hence, weak, and related/repeated)
- Can we do better?

A dream password store

- All passwords in a password store kept in *user's device or online*
  - *Using network (device or online) storage/crypto service is OK because web authentication requires data connectivity anyway*
- User memorizes a *single* master password
- Individual pwd's are *random* and *independent* of each other

A dream password store

- All passwords in a password store kept in *user's device or online*
  - *Using network (device or online) storage/crypto service is OK because web authentication requires data connectivity anyway*
- User memorizes a *single* master password
- Individual pwd's are *random* and *independent* of each other
- And: An attacker who gets hold of store and/or is in full control of the device… **still learns nothing:**
  - Adversary does not learn individual stored passwords
  - Adversary does not learn the master password

What do you mean by *nothing* ?

- Well… *nothing*. As in information-theoretic nothing!

1. Information stored in the device is *independent of the user's individual passwords and independent of the master password*
2. Attacker inside the device, w/full control, does not learn either the master password or individual passwords (not even at init!)
3. Eavesdropper or active attacker on the client-device link learns nothing

What do you mean by *nothing* ?

- Well… *nothing*. As in information-theoretic nothing!
  1. Information stored in the device is *independent of the user's individual passwords and independent of the master password*
  2. Attacker inside the device, w/full control, does not learn either the master password or individual passwords (not even at init!)
  3. Eavesdropper or active attacker on the client-device link learns nothing
- How is it possible?

  By (a form of) secret-sharing between master password and device key

SPHINX: Password **S**tore that **P**erfectly **H**ides from **I**tself (**N**o **X**aggeration) [JKSS'16, JKSS'17]

SPHINX [JKSS'17] based on Device-Enhanced PAKE [JKSS'16]

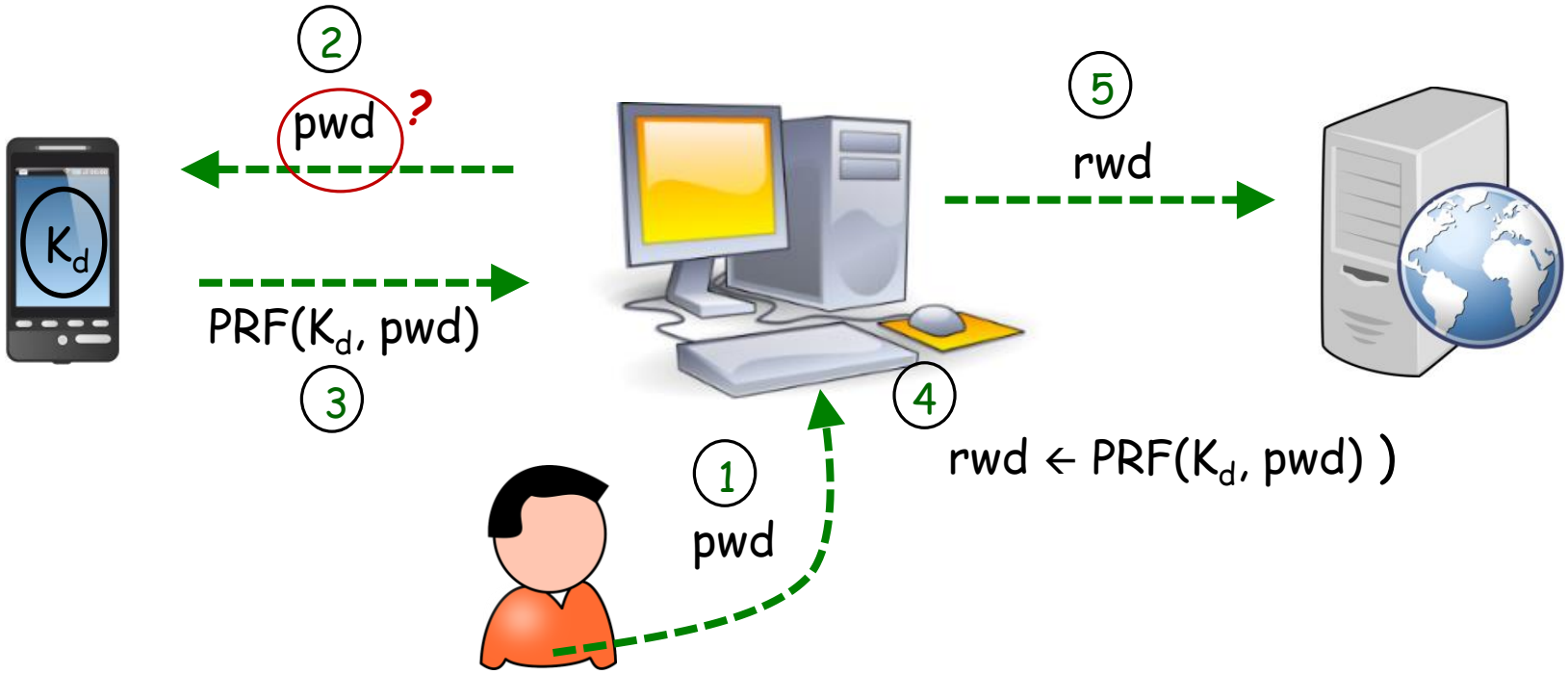
- A password Store that Perfectly Hides from Itself
- Really?

Let me show you our solution…

PRF : PseudoRandom Function
(eg. Block Cipher)

Think: PRF(K,x) = AES(K,x)

PRF-based Solution

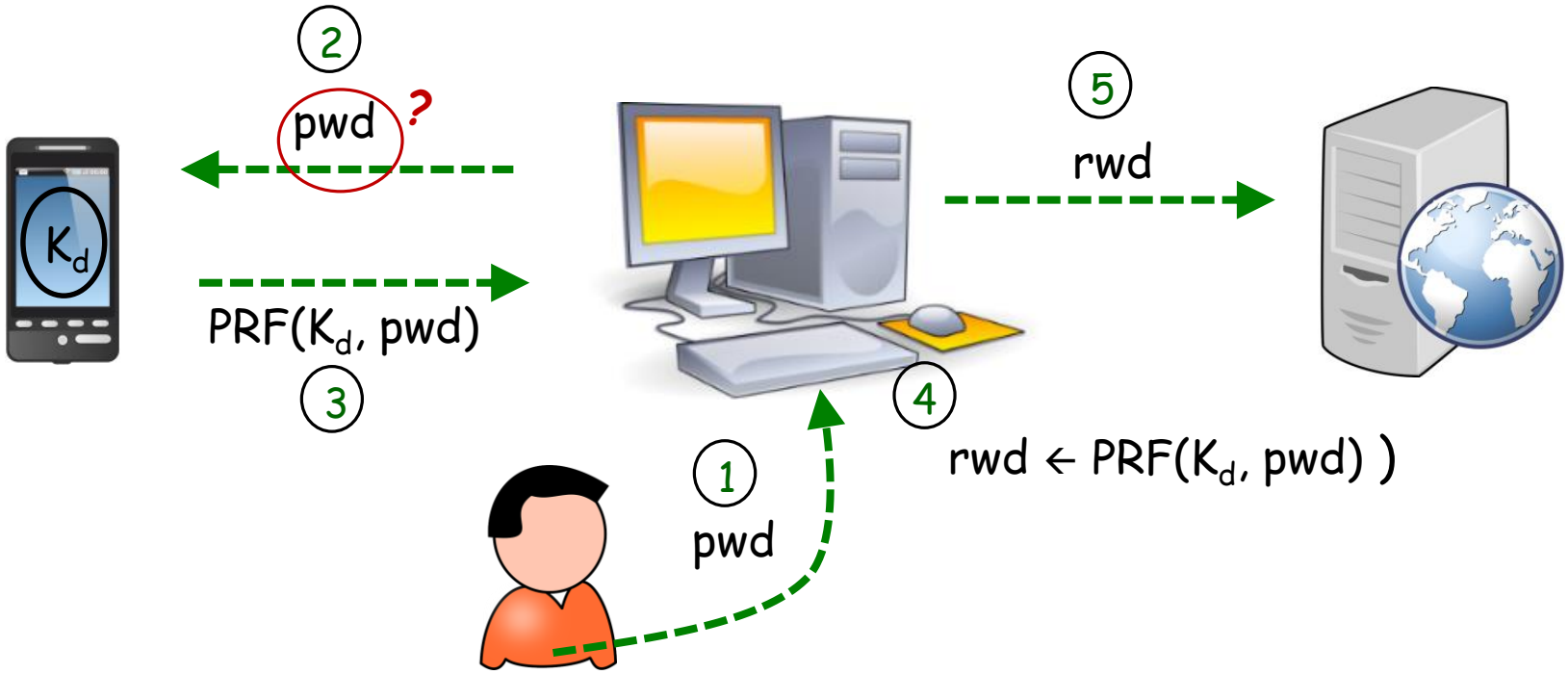


- pwd is the master password
- rwd is a (pseudo) random password that user registers with some server

PRF : PseudoRandom Function
(eg. Block Cipher)

Think: PRF(K,x) = AES(K,x)

PRF-based Solution

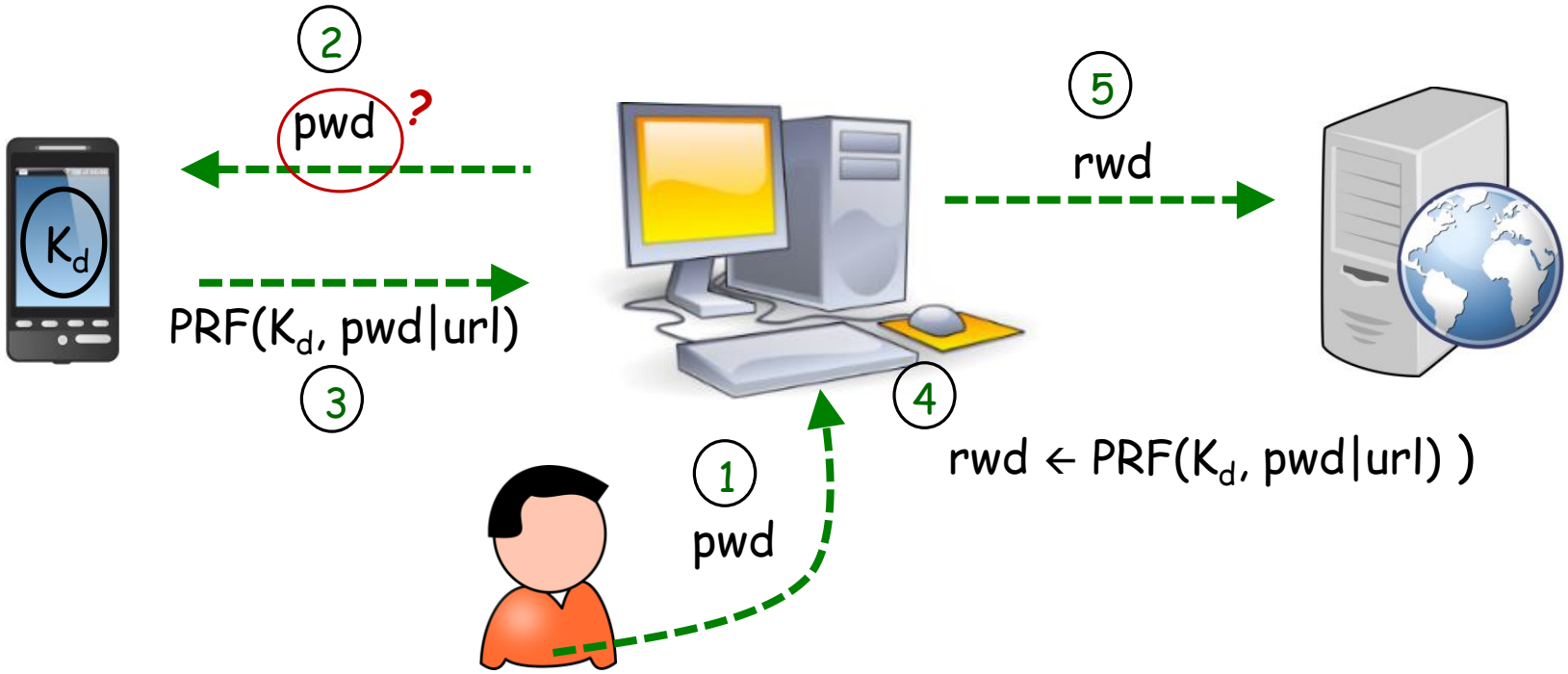


- pwd is the master password
- rwd is a (pseudo) random password that user registers with some server
- Each server has *independent* rwd, namely rwd = PRF(K_d, pwd | url)
- **Works with *any* password protocol between client and server**

PRF : PseudoRandom Function
(eg. Block Cipher)

Think: PRF(K,x) = AES(K,x)

PRF-based Solution



☺ each rwd is a (pseudo) random password → *offline attacks are infeasible*

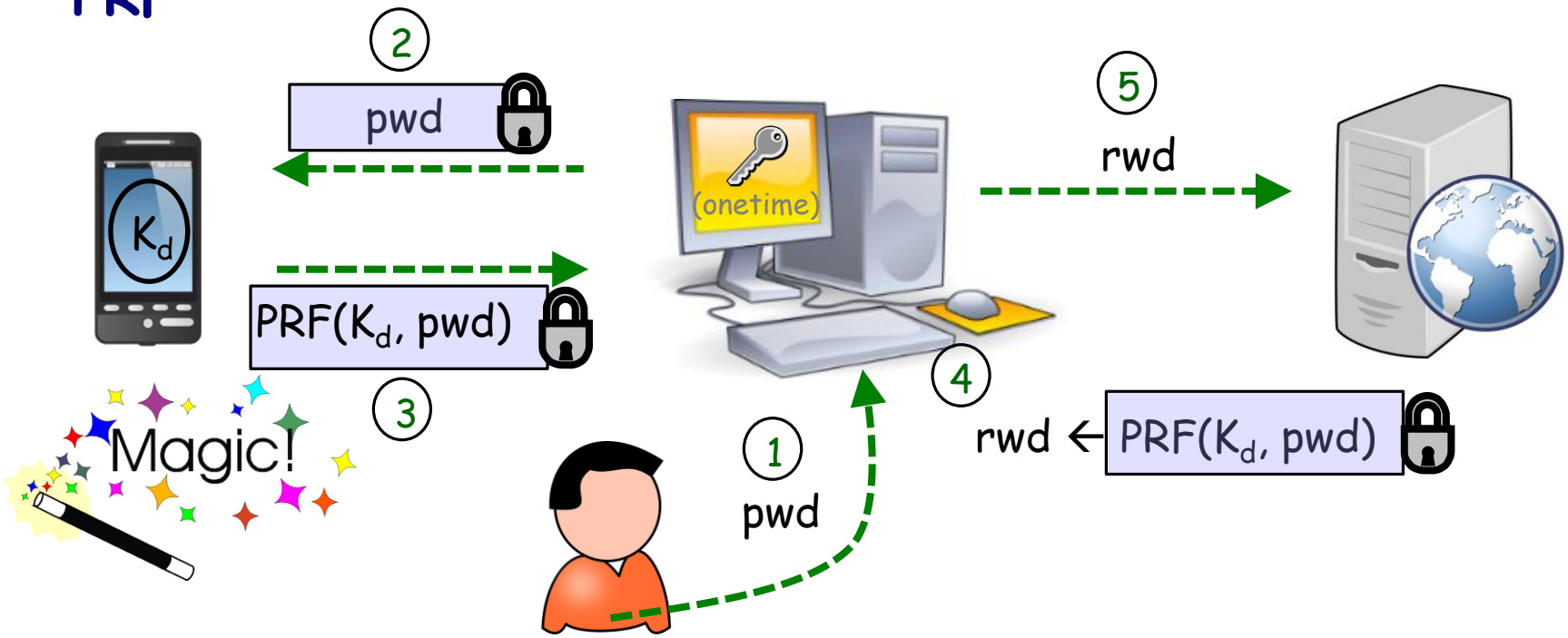
☺ storage in device (K_d) is *independent* of master pwd and of rwd's

☹ master pwd is sent unprotected to device

Oblivious PRF ← OPRF-based Solution

PRF : PseudoRandom Function
(eg. Block Cipher)

Think: PRF(K,x) = AES(K,x)



☺ each rwd is a (pseudo) random password → *offline attacks are infeasible*

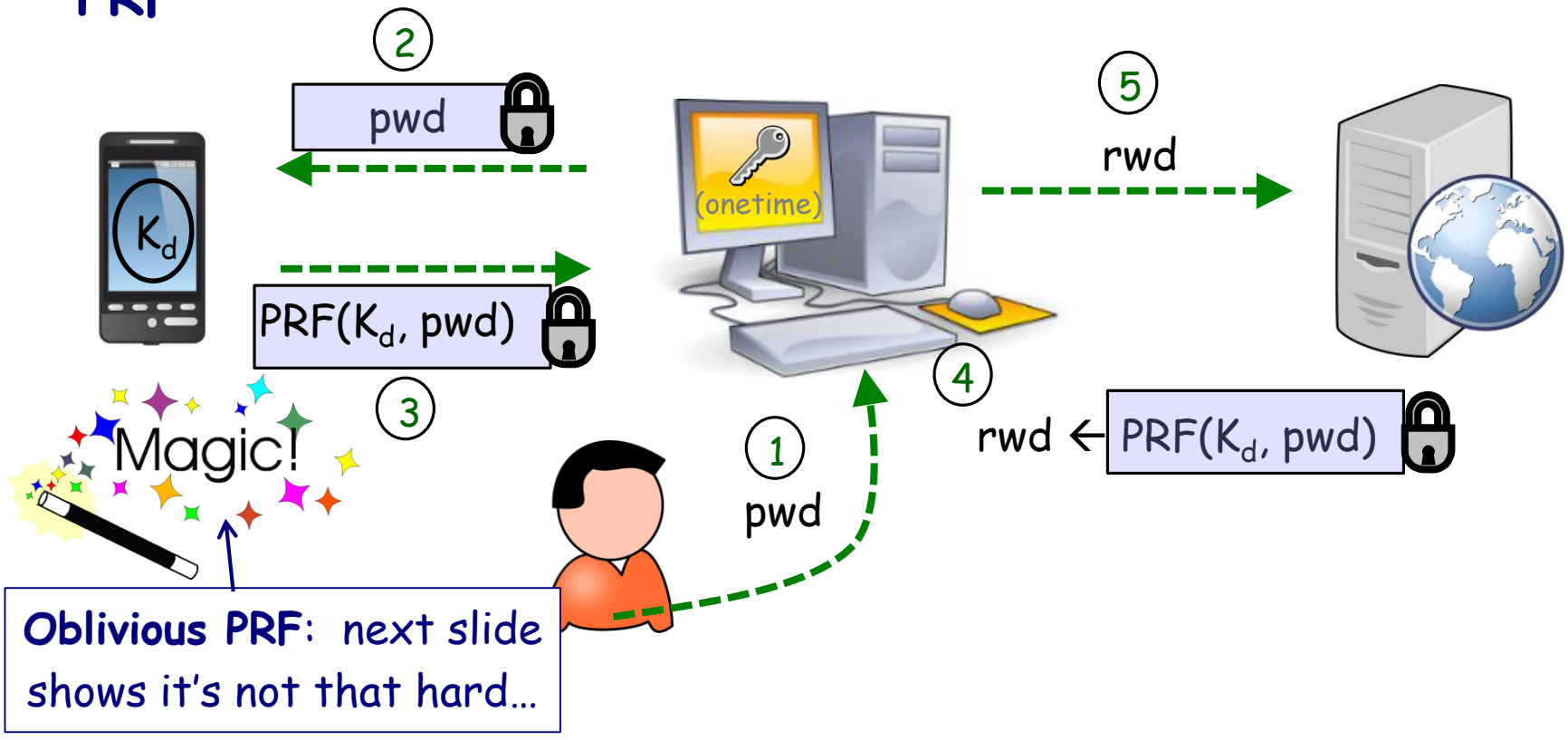
☺ storage in device (K_d) is *independent* of master pwd and of rwd's

☺ master pwd hidden over the wire and from the device

Oblivious PRF ← OPRF-based Solution

PRF : PseudoRandom Function
(eg. Block Cipher)

Think: PRF(K,x) = AES(K,x)



Oblivious PRF: next slide shows it's not that hard…

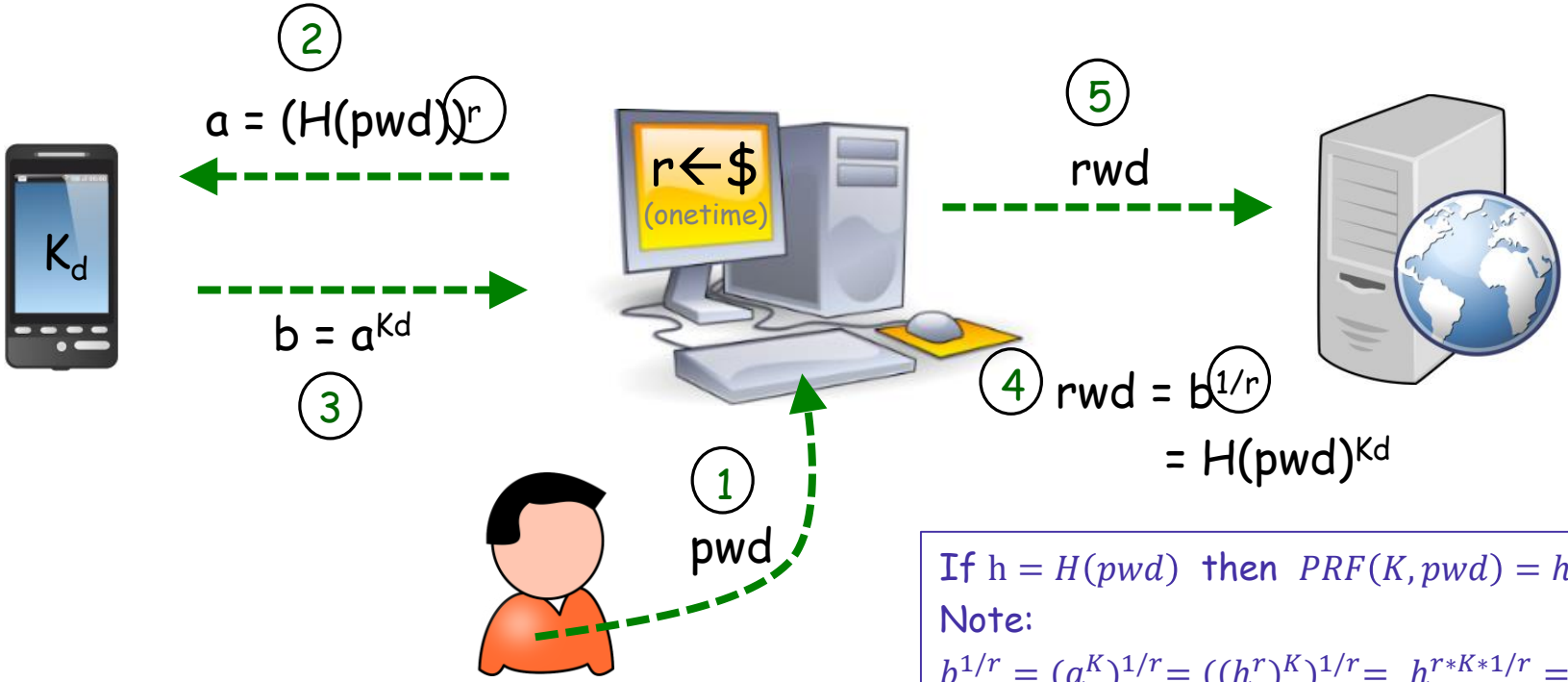
☺ each rwd is a (pseudo) random password → *offline attacks are infeasible*

☺ storage in device (K_d) is *independent* of master pwd and of rwd's

☺ master pwd hidden over the wire and from the device

Compare e.g. to RSA signature:
$sig = H(msg)^d$ for RSA private key d

Implementation: $PRF(K_d, pwd) = (H(pwd))^{K_d}$



If $h = H(pwd)$ then $PRF(K, pwd) = h^K$

Note:

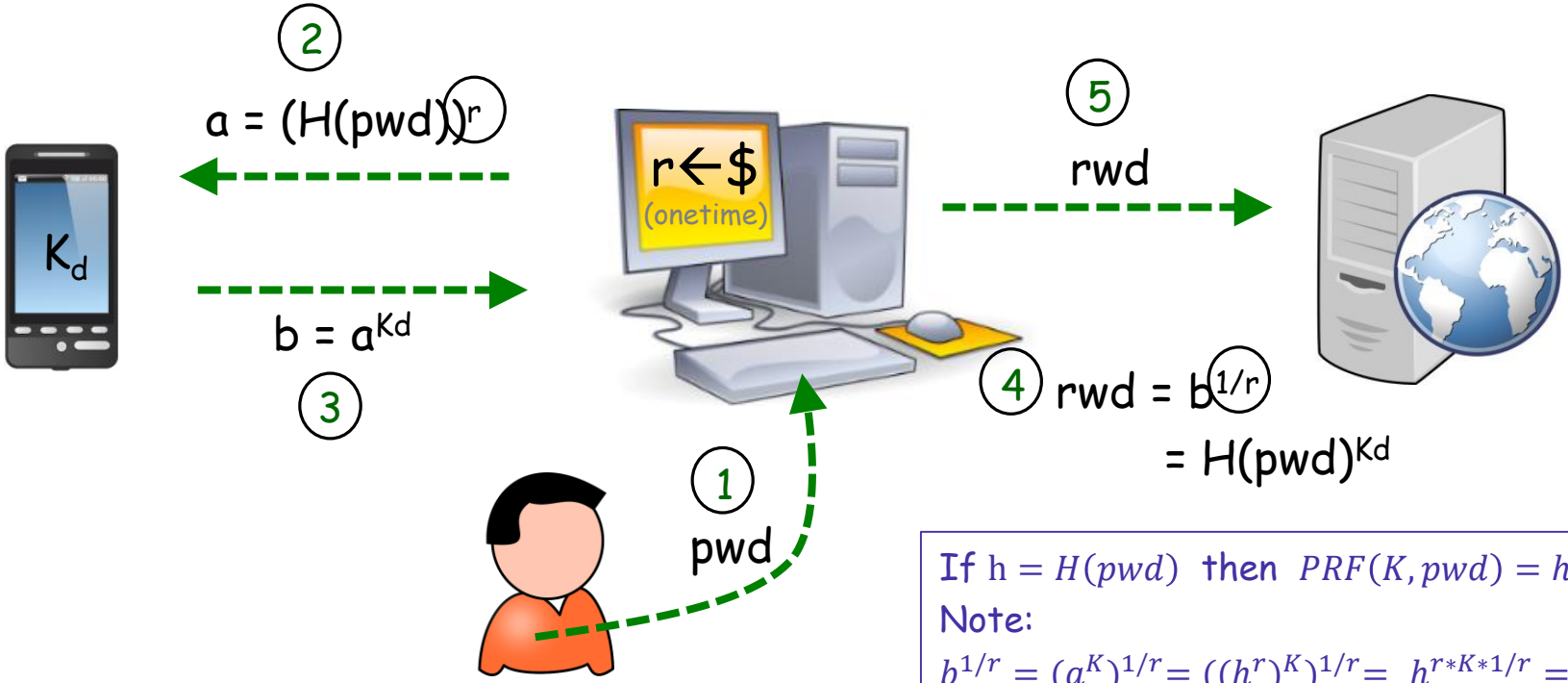
$b^{1/r} = (a^K)^{1/r} = ((h^r)^K)^{1/r} = h^{r*K*1/r} = h^K$

☺ ☺ each rwd is pseudo-random ; K_d fully independent of pwd and of rwd

☺ ☺ *master pwd is **perfectly** hidden on the wire and from device*

Compare e.g. to RSA signature:
sig = $H(msg)^d$ for RSA private key d

Implementation: $PRF(K_d, pwd) = (H(pwd))^{K_d}$



If $h = H(pwd)$ then $PRF(K, pwd) = h^K$
Note:
$b^{1/r} = (a^K)^{1/r} = ((h^r)^K)^{1/r} = h^{r*K*1/r} = h^K$

☺ ☺ each rwd is pseudo-random ; K_d fully independent of pwd and of rwd

☺ ☺ *master pwd is **perfectly** hidden on the wire and from device*

SERVER TRANSPARENT: Can use any Client-Server Password Protocol

use cases: (1) web standard "pwd-over-TLS", relies on PKI; (2) PKI-free PAKE

Not only secure…

- **Performance:** Single round C-D, 1 exponentiation for D, 2 for C, and one hash into group for C (any "DiffieHellman" group works)
  - SPHINX pwd manager: Implementation as Android app + usability study (user only inputs master pwd, rest is automated) – see references
- **Server transparent** (works with Google, Facebook, your employer…)
  - No need to protect against an eavesdropper (self-protected by SPHINX) or to authenticate user/client to device
  - Requires device authentication to the client (if Client-Server authentication protocol is PKI-based "password-over-TLS")
- ***Can replace "personal device" (cell-phone) with online service***
  - pwd, rwd never seen by server; client-to-server authentication not req'd server needs to authenticate to client (for pwd-over-tls)

SPHINX Security

- Device compromise: *unconditional* security of pwd/rwd
- Server compromise: unconditional security of pwd (and rwd)
  - Offline against master pwd ONLY if *both server and device compromised*
- Network attacks: only (unavoidable) on-line attacks
  - Against client: only if PKI keys fail
  - Against server: only if Device responds to attacker
- Client compromise: Partial defense (rwd useless in another server, master pwd useless w/o device, url hashing prevents phishing)

Two-Factor Authentication (TFA) with stronger security and usability (improves Sphinx security against Client compromise and network attacks)

Part II:

How to Protect* a *Valuable*** *Secret*

When all You Remember is a Password

*** Protect: Secrecy and Availability**

****** Bitcoin wallet, user-controlled cloud backup, secure msging keys, private key for a PK credential, corporate keys,…

How to store a secret

- Protect *secrecy* and *availability* of information while remembering a *single* password
  - Need a multi-server solution
  - Single server → Single point of failure for secrecy (offline dict. attacks) and availability (server gone → secret gone)
- Natural cryptographic solution: keep the secret encrypted in multiple locations; *secret share the encryption key* in multiple servers
  - Share among n servers, retrieve from t+1 servers (e.g. n=5, t=2)
- Protects secrecy and *availability*
  - Secret: As long as no more than t corrupted
  - Available: As long as t+1 available

Wait, but how do you authenticate to each server for share retrieval?

- Server needs to authenticate the user before delivering a share
- All we have is a user and a password
  - A strong independent password with each server? Not realistic
  - Same (or slight-variant) password for each server? Not good

➔ *Each server as a single point of failure!*

  - From one point of failure to n. We didn't achieve much, did we?

What we really want: PPSS [BJSL'11] (Password Protected Secret Sharing)

- Init: User secret shares a secret among n servers; **forgets secret** and keeps a **single password**.
- Retrieval: User contacts t + 1 servers, authenticates using the **single password** and **reconstructs the secret**.
- Security: Attacker that breaks into t servers learns nothing about secret or password
  - Even if it and finds all the server's secret information (shares, long-term keys, password file, etc)
  - Only adversary option: Guess the password, try it in an online attack
  - Offline attacks with $\leq t$ corrupted servers are useless

+ **Soundness**: User reconstructs the correct secret or else rejects

We show surprisingly efficient PPSS schemes
[BJSL'11, JKK'14, JKKX'16, JKKX'17]

- Computation:
  - Single exponentiation for each server
  - Only two exponentiations *in total* for the client (independent of t and n)
  - t multiplications (additions in ECC) for client and for each server
- Communication: Single parallel message from user to t+1 servers, one message back from each server; No inter-server communication
- *No assumed PKI or secure channels* (other than for initialization)
- Any t, n ($t \leq n$)
- Implies Threshold-PAKE (most efficient T-PAKE's to date)

Basis for Efficient PPSS Solution: OPRF

- Recall SPHINX used OPRF to transform pwd into *random secret* rwd
- To store secret x: Use OPRF to transform pwd into random rwd; store c = AuthEnc(rwd, x)
- To retrieve x: Retrieve c; use OPRF to transform pwd into rwd; set m = AuthDec(rwd, c)
- Single server solution: r = $OPRF_k(pwd) = (H(pwd))^k$ for k = server's key
- Multi-server solution: Threshold implementation of OPRF
  - n-out-of-n: Let k = k1 + k2 + … + kn

    $$OPRF_k(pwd) = (H(pwd))^{k1} \bullet (H(pwd))^{k2} \bullet \ldots \bullet (H(pwd))^{kn}$$

  - k-out-of-n: Use Shamir secret-sharing in the exponent
- Note: Can accommodate additional inputs, e.g. object/key identifier

Comparison to Prior Work (PPSS and T-PAKE)

Achieving single-round password-only protocol in the CRS and ROM models for arbitrary (n, t) parameters with no PKI requirements for any party and no inter-server communication (except for server authentication at initialization).

scheme	$(t+1, n)$	ROM/std	client	inter-server	msgs	total comm.	comp. C | S
BJKS [2]	$(2, 2)$	ROM	PKI	PKI	7	O(1)	O(1)
KMTG [6]	$(2, 2)$	Std/ROM	CRS	sec.chan.	≥ 5	O(1)	O(1)
CLN [4]	$(2, 2)$	Std/ROM	CRS	PKI	8	O(1)	O(1)
DRG [5]	$t<n/3$	Std	CRS	sec.chan.	≥ 12	$O(n^3)$	$O(1) \mid O(n^2)$
MSJ [7]	any	ROM	PKI	PKI	7	$O(n^2)$	$O(1) \mid O(n)$
BJSL [1]	any	ROM	PKI	PKI	3	$O(t)$	$8t+17 \mid 16$
CLLN [3]	any	ROM	CRS	PKI	10	$O(t^2)$	$14t+24 \mid 7t+28$
JKK'14	any	ROM	CRS	none	2	$O(t \log n)$	$2t+3 \mid 2$
	any	Std	CRS	none	4	$O(\ell t \log n)$	$O(t\ell) \mid O(l)$
JKKX'16	any	ROM	CRS	none	2	O(t log n)	t+2 | 1
JKKX'17	any	ROM	CRS	none	2	O(t log n)	2 | 1

Part III: X-PAKE*

(in preparation)

Enhanced Password Security for the Single-Server Setting (without PKI)

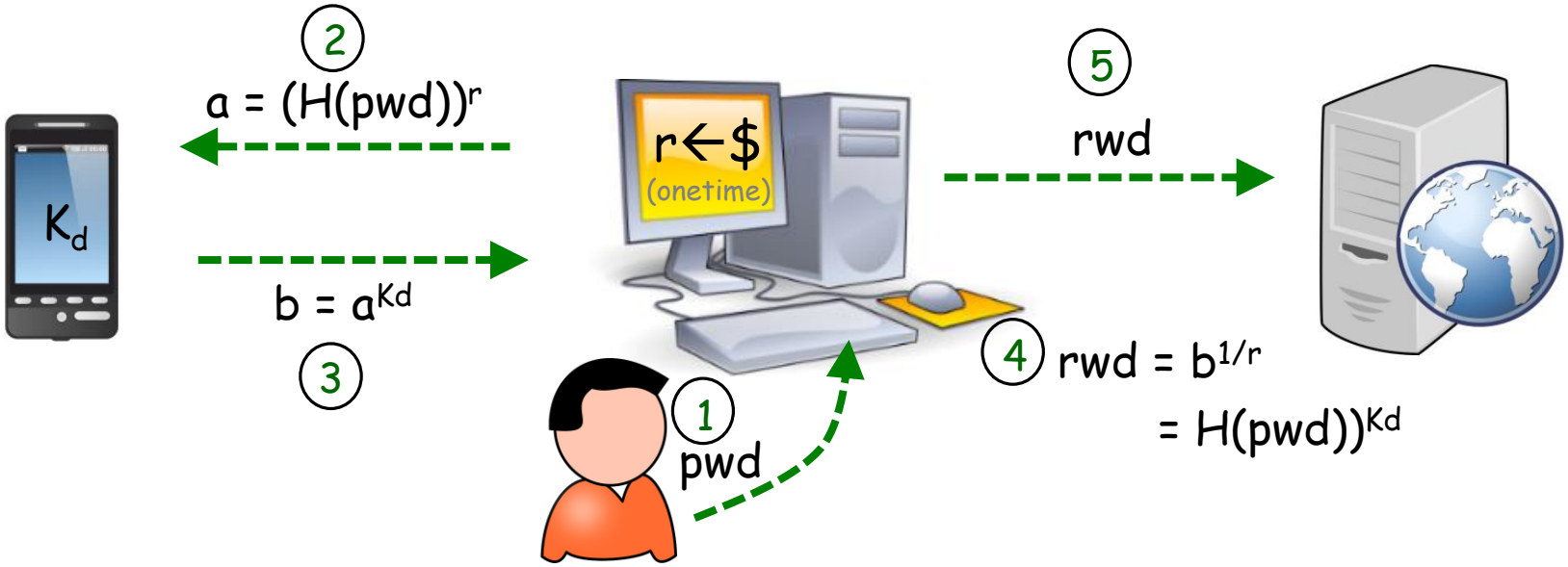
*Follows from (1,1)-PPSS, ~ Boyen'09

Single-Server PAKE
Asymmetric/Augmented PAKE (= "aPAKE")

- The goal of "asymmetric" PAKE [= client has pwd, server has H(pwd)] :
  1. Forces attacker to run a dictionary attack upon server compromise
  2. No pre-computation prior to server compromise should help
  3. Server should never see the password in plaintext
  4. Reduce/eliminate reliance on PKI
  5. Performance: Offload hash iterations to client ("key stretching")

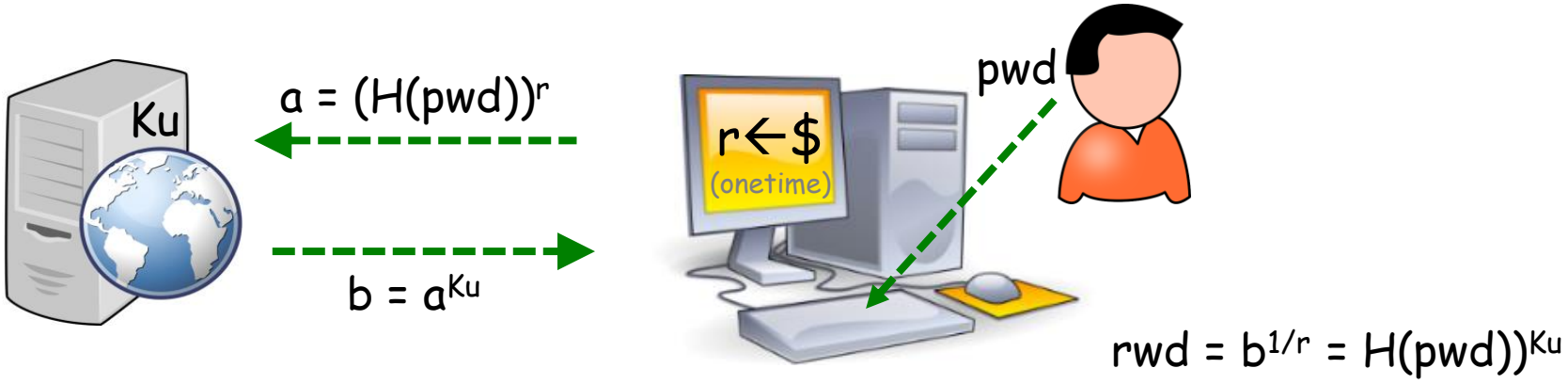
- Password-over-TLS:  1, 2  3, 4, 5
- PKI-free aPAKE:  1, 3, 4  2, 5
- **X-PAKE:**  **1, 2, 3, 4, 5**

X-PAKE: recall Sphinx/DE-PAKE [JKSS16/17]



- pwd: user's master key typed on Client ; K_d : PRF key held by Device
- rwd = PRF(K_d , pwd, url_{Srv}): pseudorandom "password" used for Server

From Sphinx/DE-PAKE to X-PAKE



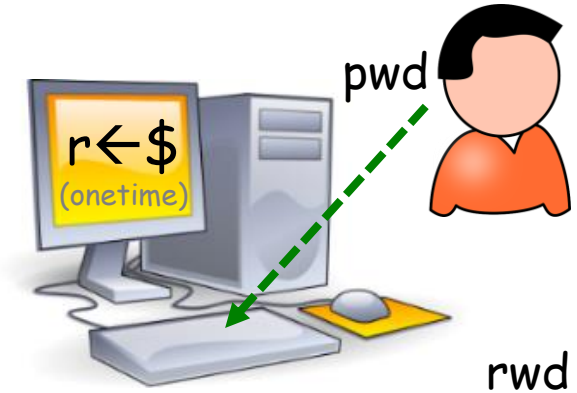
Step 1: Let Web Server run the Device (Oblivious PRF) code, denote Kd as Ku

From Sphinx/DE-PAKE to X-PAKE

Global: (PK_S, SK_S)

Per User: PK_U, Ku

$a = (H(pwd))^r$



$b = a^{Ku}$

$c = AuthEnc(rwd, (PK_S, PK_U, SK_U))$

$rwd = b^{1/r} = H(pwd))^{Ku}$

$(PK_S, PK_U, SK_U) \leftarrow AuthEnc(rwd, c)$

Step 1: Let Web Server run the Device (Oblivious PRF) code, denote Kd as Ku

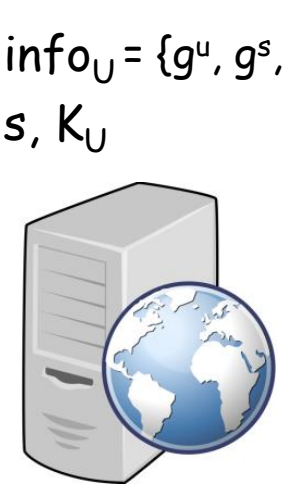
Step 2: Use rwd to bootstrap an Authenticated Key Agreement with S

- S stores $x=(PK_S, PK_U, SK_U)$ encrypted as $c=AuthEnc(rwd, x)$
- S deliver c to the client, who decrypts (PK_S, PK_U, SK_U) using rwd
- S and U run AKE on resp. inputs $(PK_S, PK_U)+SK_S$ and $(PK_S, PK_U)+SK_U$

X-PAKE with HMQV AKE (single round)

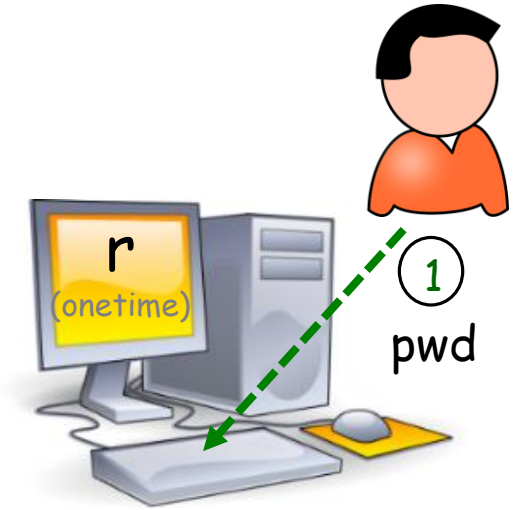
$info_U = \{g^u, g^s, c=AuthEnc(rwd, (g^s,g^u,u)), Commit(rwd,pwd)\}$

s, K_U



$U,\ a = (H(pwd))^r,\ g^x$

$info_U,\ b = a^{Ku},\ g^y$



$SK=HMQV(s, y, g^u, g^x)$

$rwd = b^{1/r}$; $(g^s,g^u,u) = AuthDec(rwd,c)$

check commitment ; $SK=HMQV(u, x, g^s, g^y)$

- x,y: one-time Diffie-Hellman exponents chosen by resp. C and S
- Single round (one message per party, add'l one for explicit auth)
- HMQV complexity (~ Diffie-Hellman KE) + 1 exp for S, 2 exp for U

X-PAKE

- X-PAKE: X is for "ECS" (Enhanced Client-Server) PAKE
- **No reliance on PKI**
- Server **never sees password**, *not even at init* (good against pwd reuse)
- **Private salt**: Attacker cannot pre-compute dictionary
  - No other PKI-free aPAKE achieves this!
- Hash **iterations can be offloaded** to user [Boyen'09]
  - No other aPAKE w/private salt (incl PKI-based) achieves this!
- Hedged PKI: **TLS-protected PAKE vs PAKE-protected TLS**
  (+ hidden pwd + offload iterations)
  - If PKI acceptable can use rwd as client signature key and TLS client auth

Summary

- Password vulnerabilities: A serious problem endangering everything from our privacy to social well-being to national security.
- Yet, we showed that ***password insecurity is not inevitable***
- "Blinded Diffie-Hellman" OPRF to the rescue in 3 applications:
  1. Password store with perfect security (device-based and/or online)
  2. Password protected secret sharing (multi-server secret protection with a single memorized password)
  3. X-PAKE: Asymmetric PAKE with extra (ordinary) properties (PKI-free, private salt, client iterated, …)
- All schemes backed by security models and proofs of security

Mature, efficient, simple technology, just waiting to be deployed…

Thanks!

Please contact us if you are interested in the prototypes of presented schemes.

Next page: referenced publications

Referenced Publications

Password Store / SPHINX / DE-PAKE:

[JKSS'16]: Jarecki, Krawczyk, Shirvanian, Saxena, **Device-Enhanced Password Protocols with Optimal Online-Offline Protection,** *AsiaCCS 2016*: 177-188

[JKSS'17]: Jarecki, Krawczyk, Shirvanian, Saxena, **SPHINX: A Password Store that Perfectly Hides Passwords from Itself**. ICDCS 2017: 1094-1104

Secret-Shared Password Store / Password-Protected Secret Sharing (PPSS)

[BJSL'11]: Bagherzandi, Jarecki, Saxena, Lu, **Password-protected secret sharing**. CCS 2011: 433-444

[JKK'14]: Jarecki, Kiayias, Krawczyk, **Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model**. ASIACRYPT 2014: 233-253

[JKKX'16]: Jarecki, Kiayias, Krawczyk, Xu, **Highly-Efficient and Composable Password-Protected Secret Sharing (Or: How to Protect Your Bitcoin Wallet Online)**. EuroS&P 2016: 276-291

[JKKX'17]: Jarecki, Kiayias, Krawczyk, Xu, **TOPPSS: Cost-Minimal Password-Protected Secret Sharing Based on Threshold OPRF**. ACNS 2017: 39-58

Two-Factor Authentication (TFA):

[SJSN'14]: Shirvanian, Jarecki, Saxena, Nathan, **Two-Factor Authentication Resilient to Server Compromise Using Mix-Bandwidth Devices**. NDSS 2014

Thank You!

Questions?

Please take our survey.

About the CTSC Webinar Series

To view presentations, join the discuss mailing list, or submit requests to present, visit:
http://trustedci.org/webinars

- *The next webinar is August 30th at 3pm Eastern*
  - *Topic: An Overview of CTSC Engagements & Application Process with CTSC's Von Welch*
- *September 25th webinar at 11am Eastern*
  - *Threat Intelligence Sharing with Romain Wartel*

The CTSC Webinar Series is supported by National Science Foundation grant #1547272.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF.

CTSC

CTSC

CENTER FOR TRUSTWORTHY
SCIENTIFIC CYBERINFRASTRUCTURE

The NSF Cybersecurity Center of Excellence

Thank You

trustedci.org

@TrustedCI

We thank the National Science Foundation (grant 1547272) for supporting our work.